# CLAIMS

1. A method of communicating between software agents in a multi-agent system, comprising the steps of:

(i) receiving, at a software agent of said system, a conversation model defining a sequence of executable tasks for implementing a role in a conversation between agents;

(ii) identifying ontology items used in the conversation model in respect of said role;

(iii) determining, for each identified ontology item, whether the software agent is operable to provide or to process the identified ontology item; and

(iv) in the event that the result of said determining step (iii) is positive, executing the conversation model to implement said role in the conversation.

2. A method as in Claim 1, wherein the conversation model includes one or more message models defining, in respect of a particular service, messages referenced in the conversation model.

3. A method as in Claim 1 or Claim 2, wherein said determining step (iii) comprises:

a) identifying at least one behaviour model which, when executed by the agent, is operable to provide or to process the identified ontology item; and

b) identifying, in respect of a particular service, a message model defining each message referenced in the conversation model.

4. A method as in Claim 3, wherein at step a), identifying said at least one behaviour comprises determining whether said at least one behaviour is operable to generate the ontology item as an output ontology item, and wherein if said at least one behaviour requires an input ontology item, determining whether the input ontology item is available in a fact base of the agent or may be produced as an output ontology item by another behaviour available to the agent.

5. A method as in any one of the preceding claims, wherein the conversation model defines an agent conversation previously unknown to the agent.

6. A method as in any one of the preceding claims, wherein the conversation model defines a plurality of roles in an inter-agent conversation, each role comprising a linked sequence of tasks which when executed by a software agent implement corresponding

stages in a conversation, and wherein a task is linked to another task in the role by means of a connector representative of either the receipt of a message from, or the output of a message to another agent implementing a complementary role defined in the conversation model.

7. A method as in any one of the preceding claims, wherein the software agent implements an initiator role defined in the conversation model and another software agent implements a responder role defined in the conversation model.

8. A method as in any one of the preceding claims, wherein a task, when executed by the agent, causes the agent to receive one or more input messages and to generate one or more output messages.

9. A method as in any one of the preceding claims, wherein at step (iv) executing the conversation model comprises selecting, for each task to be executed in respect of said role, one or more behaviours which, when executed by the agent, implement the task.

10. A software agent for use in a multi-agent system, which when executed on a computer provides:

means for receiving a conversation model defining a sequence of executable tasks for implementing a role in a conversation between software agents in said multi-agent system;

means for identifying ontology items used in a received conversation model in respect of said role;

determining means arranged, in respect of each identified ontology item, to determine whether the software agent is operable to provide or to process the identified ontology item; and

means for executing the conversation model to implement said role in the conversation in the event that the result of said determination is positive.

11. A software agent as in Claim 10, wherein said means for executing the conversation model comprise:

scheduling means for selecting a task to be executed in the conversation model; and

a task manager arranged with access to a library of behaviours to select one or more behaviours to be executed to implement the selected task.

12. A software agent as in Claim 11, wherein the software agent further comprises a fact base for storing ontology items, and wherein behaviours in said library of behaviours are arranged with access to said fact base to obtain input ontology items.

13. A software agent as in any one of claims 10 to 12, wherein the conversation model includes one or more message models defining, in respect of a particular service, messages referenced in the conversation model.

14. A software agent as in any one of claims 10 to 13, wherein said determining means are operable:

to identify at least one behaviour model which, when executed by the agent, is operable to provide or to process the identified ontology item; and

to identify, in respect of a particular service, a message model defining each message referenced in the conversation model.

15. A software agent as in Claim 14, wherein identifying said at least one behaviour comprises determining whether said at least one behaviour is operable to generate the ontology item as an output ontology item, and wherein if said at least one behaviour requires an input ontology item, determining whether the input ontology item is available in a fact base of the agent or may be produced as an output ontology item by another behaviour available to the agent.

16. A software agent as in any one of claims 10 to 15, wherein the conversation model defines an agent conversation previously unknown to the agent.

17. A software agent as in any one of claims 10 to 16, wherein the conversation model defines a plurality of roles in an inter-agent conversation, each role comprising a linked sequence of tasks which when executed by the software agent implement corresponding stages in a conversation, and wherein a task is linked to another task in the role by means of a connector representative of either the receipt of a message from, or the output of a message to another agent implementing a complementary role defined in the conversation model.

18. A software agent as in any one of claims 10 to 17, wherein the software agent, when executed, is arranged to implement an initiator role defined in the conversation model.

19. A software agent as in any one of claims 10 to 18, wherein a task, when executed by the software agent, causes the software agent to receive one or more input messages and to generate one or more output messages.

20. A software agent as in any one of claims 10 to 19, wherein said means for executing the conversation model are operable to select, for each task to be executed in respect of said role, one or more behaviours which, when executed by the software agent, implement the task.

21. A software agent as in Claim 11, wherein said scheduling means are responsive to receipt, at a message queue, of a message defined in the conversation model in respect of a task to be executed, to schedule execution of said task to be executed.

22. A method as in Claim 2, wherein said one or more message models comprise an indication of a language, ontology, rules for locating one or more recipients of a respective message, rules for creating one or more ontology items used within the contents of the message, and attributes of the message.

23. A method as in Claim 3, where in said at least one behaviour model defines one or more input ontology items, one or more output ontology items, and the location of an executable file that implements the respective behaviour.

24. A method as in any one of claims 1 to 9, wherein the conversation model, in defining a role to be executed, identifies the name of the role and, for each task defined in the role, one or more input messages to be received and one or more output messages.

25. A multi-agent system comprising a plurality of computers linked by means of a communications network, at least one of said computers being a intermediary server computer for storing conversation models in respect of one or more service providers, and wherein at least one of said computers is operable to execute a software agent as defined

according to anyone of claims 10 to 21 to implement a conversation as defined in a conversation model supplied by said intermediary server computer.